Advanced Control of Dynamic Facades and HVAC with Reinforcement Learning based on Standardized Co-simulation

Christoph Gehbauer[1*], Andreas Rippl[12], Eleanor S. Lee[1]
[1]Lawrence Berkeley National Laboratory
[2]University of Applied Sciences Technikum Wien
[*]Corresponding author. E-mail: cgehbauer@lbl.gov

Abstract

With increased complexity due to time-variable renewable electricity supply and associated variable cost, it has become evident that management of building energy demand as a resource is essential for grid stabilization. However, techno-economic and human constraints make such solutions non-trivial. Reinforcement learning (RL), a discipline of machine learning, was explored to take on this challenge. The open-source Functional Mock-up Interface - Machine Learning Center (FMI-MLC) was developed to provide a standardized interface of RL and simulation environment, through the FMI industry standard for co-simulation. RL demonstrated its ability to operate an electrochromic window and heating, ventilation, and air conditioning system (HVAC), and reduce demand during critical periods of the electric power grid.

Key Innovations

- RL, a discipline of Machine Learning which trains itself based on emulated and/or real building response, was applied to grid-interactive dynamic facade and HVAC control.
- The FMI industry standard for co-simulation, which is supported by more than 100 tools, was implemented into an RL framework to allow rapid interfacing with third-party building simulators (e.g., EnergyPlus, Modelica, Radiance).

Practical Implications

This study is a proof-of-concept for control of dynamic facades and HVAC through RL. With RL, the controller is intensively pre-trained across different climate zones in simulation, and has the capability to continue to "learn" the specific building response, once deployed. The FMI-MLC package was developed as open-source to facilitate this training process through the FMI industry standard for co-simulation.

Introduction

With global temperatures projected to rise by up to 3.2 C by 2100, aggressive goals have been set to reduce greenhouse gas emissions (GHG) by 45 % over the next decade and to net zero by 2050 (IPCC, 2018) and (DOE, 2011). The buildings sector accounts for 20 % of global demand while in the U.S. it accounts for 40 % of primary energy consumption with 51% due to heating, ventilation, air-conditioning (HVAC) and lighting loads (EIA, 2019). Of these end uses, 38 % of total commercial building electricity use and 43 % of demand during weekday peak periods are influenced by windows and the opaque envelope (Harris et al., 2019). Future building control strategies will need to consider the interdependency of demand- and supply-side technologies to minimize energy demand and associated GHG emissions.

Substantial work is underway worldwide to develop advanced control strategies. With increased digitalization of building systems, model predictive control (MPC) has become a promising control technique due to its use of analytical models and ability to optimize performance over a forecast time horizon (Drgoňa et al., 2020) and (Gehbauer et al., 2020). However, critical barriers such as lack of user-accessible modeling, commissioning, and lifetime maintenance tools will likely hinder widespread deployment. Machine learning (ML) controllers could reduce these costs significantly and accelerate deployment of adaptive, more reliable advanced controllers over the life of the installation.

In this study, we demonstrate use of an open-source toolchain to develop a ML controller. The toolchain is built on tools from the U.S. Department of Energy building energy modeling program portfolio, which enable advanced control workflows using Modelica and the FMI standard for co-simulation (Wetter et al., 2015).

The controller uses RL, a technique where a controller "learns" to map observations to actions based on a numerical reward given from an environment (Sutton and Barto, 2018). The RL controller is trained "off-site" which requires less computing power at the edge device since the computational work of training the controller is done prior to deployment. Once deployed, occupant feedback and sensor data can be used to continue training the controller for the specific building.

The objective of the RL controller is to actuate dynamic facade and HVAC systems to reduce the time-of-use (TOU) cost of electricity while maintaining occupant comfort (lighting responds to available daylight via independent photoelectric controls). Such control manages energy use based on available supply, reduces stress on the electric power grid, and as a result, enables increased adoption of renewable distributed energy resources. The RL controller was trained using feedback from a simulated office zone. The office zone was modeled using physics based models developed in Modelica and then coupled to the RL controller through the FMI for co-simulation. FMI allows the rapid replacement of training models, and since neither FMI nor RL are domain specific, in theory, the framework could be used for developing any control strategy based on RL. Several examples are given to demonstrate feasibility and performance of the RL controller compared to alternative MPC and heuristic controllers. The FMI-MLC package[1] was developed in Python, and is available as open-source on GitHub. The project page includes documentation on how to link other simulators through FMI and hosts the examples from this paper.

Methodology

This section outlines (a) the RL algorithm and applicability to controls domain, (b) the FMI that couples the building emulation model (i.e., training environment) with the RL algorithm, (c) the building emulation models, and (d) the different controller implementations. The final subsection introduces the FMI-MLC open-source software package.

Reinforcement Learning

Reinforcement learning is a discipline of ML where an artificial neural network (ANN) is autonomously trained based on a system response. Since RL is agnostic to the underlying physics and free of assumptions, it is highly applicable to many engineering problems.

The various actor-critic RL methods can be assigned to the family of off-policy algorithms which are more efficient regarding training time than on-policy algorithms. Hereby the actor constitutes the actual RL controller to be trained, and the critic constitutes an abstraction of the system response. In off-policy algorithms, observations are stored in an experience buffer and used to sample a small batch to train the agent. Experiences can be used multiple times, in contrast to on-policy algorithms where experiences can only be used once. This study uses the off-policy algorithm based on the Deep Deterministic Policy Gradient (DDPG) (Lillicrap et al., 2019) with an improvement of a n-step reward which was established with the Distributed Distributional Deterministic Policy Gradient (D4PG) (Barth-Maron et al., 2018). The n-step reward increases the consideration of long-term behavior and thus allows one to minimize the peak performance over longer horizons. The main differences to its predecessor are the continuous output values of the actor, and the introduction of target networks to calculate the action values for the actor and critic networks. The target networks are duplicates of the actual networks, but updated with a soft update, where the parameters are multiplied with a factor lower than one. This leads to an underestimation of the action value which in turn stabilizes the training process.

In this study, both the actor and critic networks are ANNs. A hyper parameter tuning process suggested best control results with three hidden layers with a size of 400 neurons for the first hidden layer and 300 neurons for the following ones for both networks. The inputs for the actor are the room temperature as state, and a four hour forecast of the outdoor air temperature, global horizontal irradiation, computed zenith angle of the sun, a schedule of occupancy, and the costs according to the TOU tariff. The outputs are the control actions for HVAC system and tint state of the dynamic facade. The critic uses the same inputs as the actor, but with additional inputs for the control actions of the actor. The output is the predicted action value.

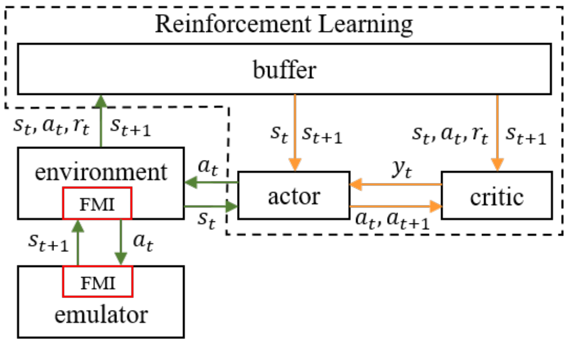

Figure 1: Interactions of the Functional Mock-up Interface - Machine Learning Center (FMI-MLC). The RL framework orchestrates the actor, critic, and training environment models. The environment is coupled through the FMI adapter to a building simulator (e.g., Modelica, EnergyPlus, etc.).

The training process, illustrated in Figure 1, starts with an exploration phase (50 episodes in this study) where actions are based purely on a noise function. After this phase, the noise starts to decline and actions from the actor become prevalent (linear decline rate to 0 at half of training episodes in this study). At this point the training of the agents is performed periodically (e.g., every 4 hours of simulation time in this study). Hereby, the actor interacts with a training environment, shown as green arrows, to select actions a_t for the current states s_t. The actions are passed through the FMI to the emulator and new

[1] https://github.com/LBNL-ETA/FMI-MLC

states are returned the same route. The results are the next states s_{t+1} and the computed reward r_t, determining if the action was a good choice. The environment response is stored in an experience buffer for future training iterations, to learn from past experiences. The training environment can be either a system emulator (i.e., building emulator as used in this study) or real system response (i.e., monitored response of a real building), or both. In the training process of the ANN, shown as orange arrows, the action value y_t is calculated with the reward r_t, sampled from the buffer and the estimation of the next action-value y_{t+1} by the critic with s_{t+1} and a_{t+1} from the actor as an input. The critic takes the state s_t and the actions a_t as inputs to estimate y_t. The actor is trained, using the estimation of the critic and trying to change the action a_t to maximize the output value of the critic.

The actions of the actor are tightly coupled with the reward function through the environment response, and therefore can be trained by maximizing rewards and minimizing penalties. To avoid convergence to local optima in the training process, the reward is not directly passed to the actor model. Instead, the sum of the immediate reward and the discounted reward of the next step are used.

The actor is trained based on the achieved reward, defined in Equations 1 to 3. The reward function represents the constraints and optimization goals which are defined by the user. In this implementation, the reward is calculated as the total costs for energy (c_E) and demand (c_D). The costs for energy and demand are normalized (n_E and n_D) between zero and one to facilitate the training process. Since the demand is charged monthly while energy cost is calculated for each training horizon (i.e., four hours in this study), the computed demand cost is scaled by s_D (i.e., 30.0 in this study) to match the magnitude of the energy cost. Penalties (p_{temp} and p_{tint}) are incurred when the room temperature (T_r) exceeds the defined boundaries (T_b), and for activation of the shading system at night. The penalties are scaled (s_{temp} and s_{tint}; 2 in this study) to avoid exceeding rewards in an effort to facilitate the training process.

$$r = -\frac{c_E}{n_E} - \frac{c_D}{n_D} * s_D - p_{temp} * s_{temp} - p_{tint} * s_{tint} \tag{1}$$

$$p_{temp} = min(|T_r - T_b|, 2) \tag{2}$$

$$p_{tint} = \begin{cases} 2 & tint < max \land Qsol = 0 \\ 0 & otherwise \end{cases} \tag{3}$$

The training process is performed for a number of episodes, until the reward converges and the behaviour of the agent is stable. In this study, the agents were trained between 1,500 to 3,000 episodes, each lasting for one day with a timestep of one hour. The length of one day was chosen to encompass variable day- and nighttime conditions, while maintaining consistency of solar loads through the day. Each episode was initialized with a random selection of the single day from all available weather data with the initial room temperature set randomly within the boundaries.

The Tensorflow and Keras packages in Python are used to construct the RL algorithm and networks. Tensorflow is capable of multiprocessing while training the ANNs with backpropagation, enabling larger batch sizes to accelerate the training process. Larger batch sizes also help to establish well trained generalized actors, an objective which is conflicting due to inherent training process of maximizing the reward.

While in a typical RL setup the environment would include the mathematical formulation of the system model (e.g., room model), this study implemented an interface instead. This interface supports the industry standard FMI and allows rapid replacement of thermal response models based on emulated or real building response.

Functional Mock-up Interface

The FMI industry standard (Blochwitz et al., 2012) describes a standardized application programming interface (API) to export and couple simulation models. A system of coupled simulation models is referred to as co-simulation. The simulation model exported in compliance with FMI is referred to as a functional mock-up unit (FMU). It contains the files below in a zip file ending with $.fmu$:

- An Extensible Markup Language (XML) file describing parameters, inputs, outputs, and dependencies of the model.
- Compiled C-code with standardized FMI functions to evaluate the model.
- Resource data which can contain additional information such as documentation, dependency files that are required by the simulation, graphical illustrations, or source code.

The FMI standard defines two types of simulation model export. The co-simulation (CS) export contains the numerical solver, stores internal states, and provides the integrated result for the next timestep, i.e., start time plus step size as an output. With CS the models are forced to advance with a predefined step size. The model exchange (ME) export contains the system of equations and requires an external solver. While the external solver requirement imposes additional dependencies it solves coupled models as a system, i.e., in a single-pass. CS is often preferred for single or a low number of models, while ME is superior for large interconnected systems. Coupled models can be solved with an orchestrator to coor-

Table 1: Overview of emulators

	simple	complex
Size	13.9 m^2-floor; 5.2 m^2-window	
Window	double-pane electrochromic	
Equipment	10.8 W/m^2; scheduled	
Occupancy	100 W; scheduled	
Lighting	5.0 W/m^2; dim to 350 lx	
HVAC	$COP_{cool} = 3.5$; $COP_{heat} = 4.0$	
Heat balance	RC response	Modelica
Solar loads	simple (SHGC)	Modelica

dinate the evaluation of models and interfacing with a global solver. The orchestrator used in this framework, PyFMI (Andersson et al., 2016), provides the capability to interface with both model types. However, all FMUs in this study were exported with the CS standard which simplifies portability and comparability.

Using the standardized FMI to interface the training environment with the reinforcement algorithm offers multiple benefits: (a) leveraging an agreed-upon industry standard which is well maintained, updated, and improved by industry stakeholders, (b) the ability to utilize a large variety of model libraries for different domains that are built and maintained by industry and research institutions, (c) the flexibility to couple models of different domains, e.g., thermal heat balance envelope model with detailed daylight models to form a single multi-domain co-simulation, (d) industry developed wrappers for the FMI standard in different programming languages (e.g., Python, MATLAB, Modelica, Java) that are well maintained and documented, and (e) a large community working on FMI and related standards.

Building Emulation Models

Two building emulation models were developed and are summarized in Table 1: (a) a simple emulator based on a simplified Resistance-Capacitance (RC) model and reduced-order analytical models, and (b) a complex emulator based on a building physics heat balance model. Both models were developed in Modelica and interfaced through the FMI. The simple emulator was used for proof-of-concept and to establish hyper parameters and forecast horizon of the RL setup, while the complex emulator was used to demonstrate applicability on a realistic building response.

Both emulators represent a perimeter office with a floor area of 13.9 m^2 and window area of 5.2 m^2 (window-to-wall ratio of 33 %). The office was modeled to be adjacent to other offices with indoor surfaces modeled as adiabatic. The only surface allowing heat exchange was the exterior, south-facing wall. The total resistance value (R-value) of the exterior wall was calculated by applying the U-values (0.32 and 1.6 W/m^2-K, respectively) and respective wall- and window areas. The total capacity of the room was calculated based on the air volume and an estimation of effective thermal mass of all surfaces based on EN-ISO 13786. The exterior wall was modeled as lightweight construction, ceiling and floor were modeled with concrete construction and all interior walls were gypsum wallboard. The total R-value of the room was 0.085 K/W with a capacitance of 2,029 kJ/K. The internal loads included scheduled office equipment of 10.8 W/m^2 and scheduled occupant thermal loads of 100 W. The office was occupied from 07:00 to 18:00.

Depending on the controller evaluated, the HVAC system was either controlled by the heating or cooling output, or could be set to an independent internal control loop. This internal control loop represent a Proportional Integral (PI) thermostat control. The maximum thermal power for heating and cooling was limited to 75 W/m^2. To compute energy use, a reversible heat pump system using a fixed coefficient of performance of 3.5 for cooling and 4.0 for heating was used.

The dynamic facade was a dual-pane electrochromic (EC) window. The EC window has four discrete tint states with a visible transmittance (Tvis) between 60 % (clear), 18 %, 6 %, and 1 % (dark) and solar heat gain coefficient (SHGC) between 42 % (clear), 16 %, 12 %, and 10 % (dark). In the simple emulator, solar heat gains were calculated using global and diffuse horizontal irradiation, and direct normal irradiation from typical meteorological year (TMY) weather data, longitude and latitude, and orientation and tilt of the window. The Tvis and SHGC values given above for normal incidence were used for the calculation. In the complex emulator, solar heat gains were computed using the Modelica window heat balance model using the actual angle-dependent solar-optical properties of the EC window.

The lighting system was modeled with an installed power density of 5.0 W/m^2, scheduled occupancy-based control, and dimming control to maintain an work plane illuminance (WPI) of 350 lx based on available daylight. Daylight illuminance was calculated using daylight factors[2] in both the simple and complex emulators.

Both models used the Pacific Gas and Electricity E-19 TOU electricity tariff (PG&E, 2020) for calculating energy and demand costs.

Controllers

All controllers were designed to minimize the cost of energy and demand, and maintain room temperature and WPI within defined limits through active control of the EC window and HVAC systems. Within RL, objectives and constraints are described in a single reward function (Equation 1). The RL controller

[2]https://www.uk-ncm.org.uk/filelibrary/SBEM-Technical-Manual_v5.2.g_20Nov15.pdf

used an external weather and load forecast of 4 h, and internally predicted another 4 h, which resulted in a total horizon of 8 h for all cases. The 4 h forecast and 8 h horizon were established through a hyper parameter search where it resulted in the optimal balance between ANN network size and corresponding training time and the optimal control result. A MPC controller was implemented as a gold-standard controller with weather, occupancy, internal loads, RC parameters and the electricity tariff information that matched the simple emulation model. The optimization goal and constraints of the MPC controller were the same as for the RL controller, but were formulated as mathematical constraints and objectives, rather than rewards. The optimization horizon of the MPC controller was 24 h. Note that the RC parameters of the MPC controller did not match those of the complex emulation model because building assembly construction descriptions, not RC values, were modeled in the complex emulated environment. Initial RC values were based on the the EN-ISO 13786 and were manually updated to better match the response of the complex emulator. The MPC controller therefore represents a realistic implementation where exact parameters for RC are unknown[3]. Both, the RL and MPC controller were evaluated with and without HVAC control. In the case of independent HVAC control, an internal PI controller was used through thermostat control. In addition, a simple Heuristic controller was implemented to represent a typical installation where EC control was independent from HVAC control. The Heuristic controller tints the EC window based on computed incident solar irradiation.

FMI-MLC Package and Code Example

The open-source FMI-MLC package[1] is designed to mask most of the complex interactions through high-level functions and classes. FMI-MLC was built as a generic tool to simplify the development of RL controllers by relying on the standardized FMI and Tensorflow's py_environment to link the training environment. At its current stage of development, any thermal simulator is supported as a training environment as long as it provides a control signal as input and temperature response as output. Simulation parameters of any type can be set through FMI-MLC and are passed to the emulator through the FMI. To illustrate the functionality of FMI-MLC, a simple example is given to initialize the simple RC model and RL controller, and to conduct the training process.

The *fmi_mlc.env* extension contains the environment agent shown in Figure 1, with the FMI adapter to link to the emulator. The path to the FMU is given as *fmu_path*. Parameters of the FMU can be modified through the *env_par* dictionary. The *fmi_mlc.env* extension also allows the setting of solver options of the FMU, and variable names of inputs and outputs to communicate with the environment and further with the RL framework. The default variable names (e.g., *Q* for heat input, *T_out* for outdoor temperature, *T_in* for resulting indoor temperature) can be mapped to those of a custom FMU using the *par* flag of *fmi_mlc.env*. This mapping is usually the only alteration needed to link any FMU to the FMI-MLC package. Similarly, the *fmi_mlc.wrapper* provides many configuration options to tune hyperparameters of the learning process and specify agent network structures. Full documentation is available through the project site.

```
# Import FMI-MLC package
import fmi_mlc
# Configure environment
fmu_path = 'fmus/linux/RCmodel.fmu'
env_par = {'R': 0.085, 'C': 2.029e6}
env = fmi_mlc.env(env_par = env_par,
                  fmu_path = fmu_path)
# Initialize agents
agents = fmi_mlc.wrapper(env = env)
# Train agents
agents.train()
# Print reward
print(agents.actor.res['reward'])
```

Results

In order to demonstrate the applicability of RL for grid-interactive dynamic facade and HVAC control, two proof-of-concept test cases are given. The first case compares a RL and Heuristic controller with MPC to evaluate performance in a hypothetical setup (i.e., simple emulator). This illustrates a real-world installation where the RL controller is pre-trained based on 15 climate zones (i.e., IECC climate zones 1 through 7), and then virtually deployed to a specific site (in both cases using the simple emulator). The second case is similar to the first one, but instead uses a RL controller trained and evaluated on the complex emulator to illustrate its performance with a realistic building response. Test cases are evaluated for the 3B coastal climate zone (Los Angeles, CA) using one sunny day (July 31st) and one cloudy day (July 29th), and the full month of July.

The FMI-MLC and Tensorflow Python packages are available for most modern computing platforms. In this study, off-site pre-training was performed on a workstation with Intel Xeon W-2145 CPU with 3.70 GHz and 64 GB memory, without GPU acceleration. Tensorflow was set to use all of the 16 available CPU cores. Computing power and memory will impact RL controller training and convergence times. Note that once the RL controller is trained off-site (i.e., through simulation), the hardware requirements are much reduced where embedded controllers such as a Rasp-

[3]The complex emulation model represents a realistic building response which can only be approximated with a single RC network.

berry Pi[4] are sufficient for deployment in a building.

Simple Emulator

The first evaluation provides a comparison between a RL controller to a perfect information MPC. Note that a realistic MPC implementation would not have perfect values for R and C, and therefore would perform worse than this technical optimum. Heuristic control is provided as a state-of-the-art benchmark. The training dataset included annual TMY weather data for 15 climate zones. All control strategies were evaluated with the simple emulator for the 3B coastal climate. The test days used for evaluation (July 29th and July 31st) were removed from the training dataset. The monthly evaluation of July consisted of mainly in-sample weather data, except the removed days.

Table 2: Results from the simple emulator with RL, MPC, and Heuristic control in July, climate zone 3B coastal. Critical demand period from 12:00 to 18:00.

	Month	Sunny	Cloudy
	Total Cost [$]		
MPC (EC&HVAC)	21.0	10.9	10.3
RL (EC&HVAC)	24.8	14.2	13.9
Heur. (EC only)	27.5	15.4	14.5
	T-penalty [Kh]		
MPC (EC&HVAC)	0.0	0.0	0.0
RL (EC&HVAC)	0.8	0.1	0.1
Heur. (EC only)	0.0	0.0	0.0
	Critical Demand [W]		
MPC (EC&HVAC)	268.1	246.8	232.6
RL (EC&HVAC)	304.7	303.6	303.9
Heur. (EC only)	348.5	349.8	330.4

The RL controller was pre-trained with 1,500 episodes (5,900 ANN updates) before the virtual deployment. The total training process took 4:24 hours with the specified hardware. However, the RL controller was able to maintain the room temperature most of the time after 200 episodes (total runtime of only 7 minutes), with total energy costs equivalent to that of Heuristic control. The results of the controller evaluation are shown in Table 2.

The total monthly cost of RL control, which includes energy and demand costs, was 18.1 % higher than the MPC case. The monthly cost of Heuristic control was 31.0 % higher than the MPC case. While Heuristic control relied on the independent room thermostat to keep temperatures within bounds, RL and MPC were able to actively control the HVAC system while maintaining acceptable room temperature. In the RL case, the temperature was slightly out of bounds (i.e., about 0.1 K for a single 1 hour timestep) some days in the late afternoon. The critical demand was calculated as maximal hourly demand during the critical on-peak period from 12:00 to 18:00. RL control increased critical demand by 13.7 % in comparison with the MPC case, but reduced demand by 14.4 % in comparison with Heuristic control.

[4]https://www.raspberrypi.org/

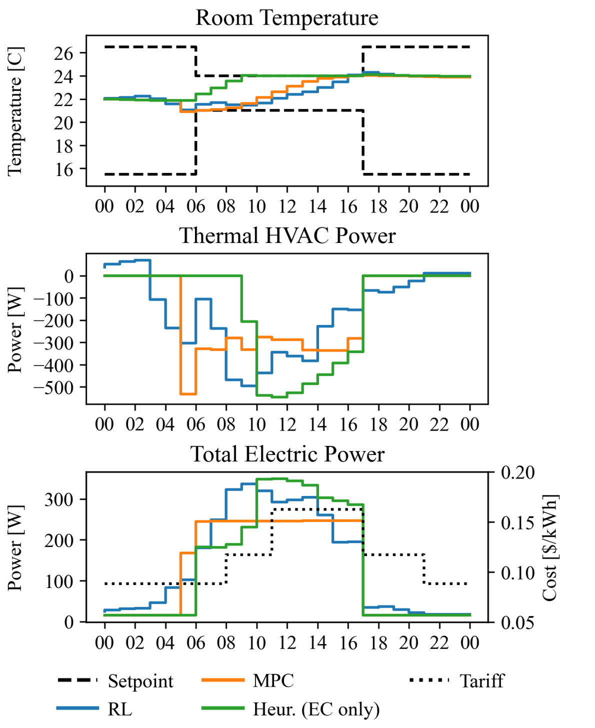

Figure 2: Results for RL, MPC, and Heuristic controller with simple emulator on a sunny day in July.

Figure 2 compares the room temperature, HVAC power, and resulting total electric power consumption between the three control cases. As shown in the middle plot, the RL controller pre-cools the building (negative HVAC power reflects thermal cooling of zone) starting from 04:00, which is 2 hours earlier than MPC control. Due to the earlier onset of precooling the RL controller can use less cooling power in the early morning hours to closely match the temperatures of the MPC case, shown in the upper plot. While both RL and MPC keep temperatures close to the lower temperature bound (i.e., heating setpoint) for most of the morning hours, the Heuristic controller does not control the HVAC system so the temperature floats with independent thermostat control until the upper temperature bound (i.e., cooling setpoint) is reached at 10:00. Starting from 10:00 RL and MPC, on the other hand, slowly release the thermal mass, effectively reducing the required HVAC power as indicated by the reduced cooling demand during afternoon hours. With this strategy, the peak cooling load for Heuristic control occurs during the high priced peak period which leads to the highest costs in this comparison. The lower plot shows the total electric power demand of the room. This includes HVAC, internal equipment, and lighting power. At onset of the high-priced on-peak period, from 12:00 to 18:00 (tariff denoted on the second y-axis in the lower plot), MPC maintained an optimal constant load of 246.8 W, RL reached a peak demand of 303.9 W, and Heuristic of 349.8 W, for the sunny day in July. Mismatch between the RL controller and the simple emulator caused the RL controller to underestimate available thermal mass. This was compensated for by a small heating input in the early morning hours to avoid a penalty for violating thermal comfort.

Complex Emulator

The second case is similar to the first one, but uses the complex emulator for both training of the RL controller and evaluation of all cases. Training included the 15 climate zones. It represents a more realistic implementation where none of the controllers can achieve the technical optimum. Heuristic was evaluated for EC only control, RL and MPC were evaluated for both, EC only and combined EC and HVAC control.

Table 3: Results from the complex emulator with RL, MPC, and Heuristic control (with and without HVAC control) in July, climate zone 3B coastal. Critical demand period from 12:00 to 18:00.

	Month	Sunny	Cloudy
	Total Cost [$]		
RL (EC&HVAC)	29.4	16.9	16.9
MPC (EC&HVAC)	26.9	15.4	14.4
RL (EC only)	27.6	16.0	14.9
MPC (EC only)	27.7	16.2	14.0
Heur. (EC only)	29.1	16.5	14.4
	T-penalty [Kh]		
RL (EC&HVAC)	3.4	0.5	0.8
MPC (EC&HVAC)	21.7	2.5	0.4
All EC only cases	0.0	0.0	0.0
	Critical Demand [W]		
RL (EC&HVAC)	381.3	389.0	391.0
MPC (EC&HVAC)	358.2	349.8	320.7
RL (EC only)	386.7	369.9	345.2
MPC (EC only)	390.8	373.2	320.7
Heur. (EC only)	398.5	379.7	329.6

Figure 3 shows a subset of controller cases for the complex emulator. The RL for EC and HVAC control (blue) pre-cools the building in the morning hours but mismatch in the controller leads to overestimation of thermal mass and increased HVAC load during the high-priced peak period, from 12:00 to 18:00. Starting from 14:00 thermal mass is released and RL for EC and HVAC shows lowest HVAC loads. MPC for EC and HVAC (green) does not utilize pre-cooling and slowly releases thermal mass through the day. Model mismatch leads to slight temperature deviation from the cooling setpoint starting at 13:00. However, this deviation reduces HVAC load and leads to lowest total electric power consumption. RL for EC control only (orange) cannot control the HVAC system and quickly reaches the cooling setpoint where it remains for the day.

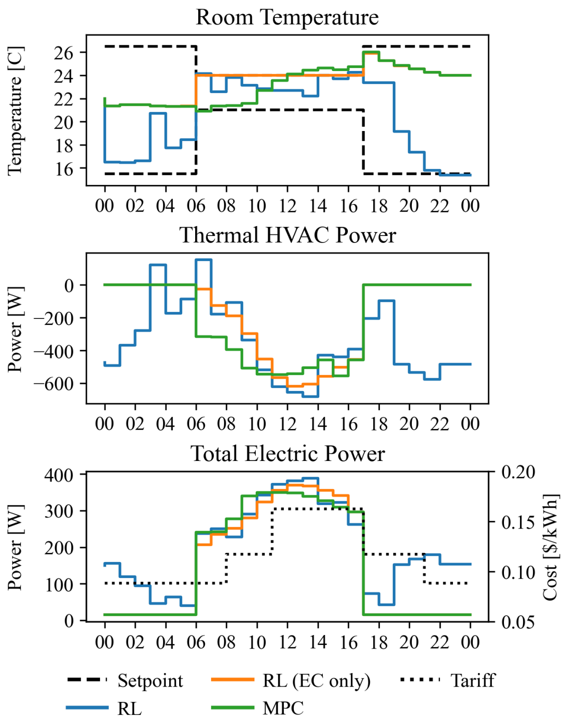

Figure 3: Selected results for RL and MPC controller with complex emulator for a sunny day in July.

While the controller with lowest monthly total energy cost was MPC for EC and HVAC control (Table 3), it resulted in the highest temperature deviations of about 0.5 K for 4 hours on most days in the afternoon. The lowest cost without temperature deviation could be achieved by RL for EC window control only. Inclusion of HVAC in RL control did not lower cost as one would expect. Instead, it raised monthly costs from $ 27.6 to $ 29.4, which is likely due to overfitting of the RL model. Further adjustments to training parameters are needed to improve RL estimation of thermal mass effects. On the other hand, RL for EC and HVAC control lowered critical demand from 27.8 to 27.4 W/m^2, indicating that monthly cost differences were incurred during non-peak periods. Heuristic control resulted in the highest cost for EC only cases, with a critical peak demand of 28.7 W/m^2.

Discussion

The control of dynamic facades is a challenging problem considering the interactions with occupant, lighting and HVAC systems, and building-level power demand (energy and peak demand charges). Technologies to optimally balance the control between these (conflicting) objectives are in development, but typically require complex commissioning. One example would be MPC, where the building is internally represented as a RC thermal model with various interactions with the environment. Setting up such an RC model is non-trivial and requires data inputs of thermal and electrical loads. However, a well-trained MPC controller can achieve a reasonable optimum. On the other hand, RL has a much reduced commissioning time. The RL is generically pre-trained using either large datasets from real buildings, or building emulators with different construction and weather data, as used in this study. Once a certain performance is reached in-silico, the RL controller is ready to be deployed in any building. While not explored in this study, the RL controller can continue to learn and specialize in the specific building deployed.

The RL controller showed very good performance (i.e., 9.8 % reduced cost and 12.6 % reduced critical demand in comparison to Heuristic control) with the simple emulator, and lowest cost for EC only control with the complex emulator. While those cases were simplified proof-of-concept implementations with perfect knowledge of weather and occupancy forecast, the feasibility of RL control as a generic controller for any building (without commissioning) was demonstrated. Further research will be necessary to thoroughly identify the role RL control can play in grid-interactive building control.

While RL can provide the discussed benefits, it also struggles with complicated or conflicting objectives, as indicated by the poorer performance when using RL for EC and HVAC control, in comparison to EC control only, in the complex emulator case. We believe that more specialized models such as Long-Short Term Memory (LSTM) could be integrated with traditional ANNs to improve upon that. But in a short run, the conclusion might be to use this technology for dynamic facade control only or to implement a hierarchical structure and define control limits.

Conclusion

Reinforcement learning poses a tremendous potential for dynamic facade control. An early proof of concept, where electrochromic windows and HVAC were controlled to reduce zonal electricity cost, while maintaining occupant comfort, was demonstrated. The flexibility of the FMI industry standard for co-simulation was exploited to rapidly change between different model setups. With the ever increasing computational resources, it will be possible to develop reinforcement learning controllers with more complex considerations, such as occupant glare constraints and diverse building types.

Acknowledgment

This work was supported by the Assistant Secretary for Energy Efficiency and Renewable Energy, Building Technologies Office, of the U.S. Department of Energy under Contract No. DE-AC02-05CH11231. This work was also supported by the Austrian Marshall Plan Foundation.

References

Andersson, C., J. Åkesson, and C. Führer (2016). *Pyfmi: A python package for simulation of coupled dynamic models with the functional mock-up interface.* Centre for Mathematical Sciences, Lund University.

Barth-Maron, G., M. W. Hoffman, D. Budden, W. Dabney, D. Horgan, et al. (2018). Distributed distributional deterministic policy gradients. *arXiv preprint arXiv:1804.08617*.

Blochwitz, T., M. Otter, J. Akesson, M. Arnold, C. Clauss, et al. (2012). Functional mockup interface 2.0: The standard for tool independent exchange of simulation models. In *Proceedings of the 9th International MODELICA Conference; September 3-5; 2012; Munich; Germany*, pp. 173–184. Linköping University Electronic Press.

DOE (2011). Buildings energy data book. *Department of Energy*.

Drgoňa, J., J. Arroyo, I. C. Figueroa, D. Blum, K. Arendt, et al. (2020). All you need to know about model predictive control for buildings. *Annual Reviews in Control 50*, 190 – 232.

EIA (2019). International energy outlook 2019. *U.S. Energy Information Administration*.

Gehbauer, C., D. H. Blum, T. Wang, and E. S. Lee (2020). An assessment of the load modifying potential of model predictive controlled dynamic facades within the california context. *Energy and Buildings 210*, 109762.

Harris, C., S. Mumme, M. LaFrance, M. Neukomm, and K. Sawyer (2019). Grid-interactive efficient buildings technical report series: Windows and opaque envelope.

IPCC (2018). Special report on the impacts of global warming of 1.5 c above pre-industrial levels and related global greenhouse gas emission pathways. *Intergovernmental Panel on Climate Change*.

Lillicrap, T. P., J. J. Hunt, A. Pritzel, N. Heess, T. Erez, et al. (2019). Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*.

PG&E (2020). Electric schedule e-19 medium general demand-metered tou service, effective april 19 2020. *Pacific Gas and Electric*.

Sutton, R. S. and A. G. Barto (2018). *Reinforcement learning: an introduction.* The MIT Press.

Wetter, M., T. S. Nouidui, D. Lorenzetti, E. A. Lee, and A. Roth (2015). Prototyping the next generation energyplus simulation engine. In *Accepted: 13-th IBPSA Conference. International Building Performance Simulation Association.*